

Algoritmos y Estructuras de Datos II

Primer parcial — Viernes 29 de abril de 2022

A

1	2	3
P	P-	A

Corrector:

Luis

- Es posible tener una hoja (2 carillas), escrita a mano, con los apuntes que se deseen
- Cada ejercicio debe entregarse en **hojas separadas**.
- Incluir en cada hoja el número de orden asignado, número de libreta, número de hoja, apellido y nombre.
- Cada ejercicio se calificará con Perfecto, Aprobado, Regular, o Insuficiente.
- El parcial estará aprobado si las notas de los tres ejercicios son mejores o iguales a alguna de las siguientes combinaciones: $\{R, A, R\}$, $\{R, R, A\}$ o $\{I, A, A\}$.
- Los ejercicios **no** se recuperan por separado.

Ej. 1.

Axiomatizar la función **tieneValor** que, dado un diccionario a naturales y un natural, establece si el diccionario tiene alguna clave cuyo significado sea el segundo parámetro. Esto es:

$$\text{tieneValor} : \text{dicc}(\alpha, \text{nat}) \times \text{nat} \rightarrow \text{bool}$$

Que cumple que:

$$(\forall a : \text{nat}, d : \text{dicc}(\alpha, \text{nat})) \text{ tieneValor}(d, a) \Leftrightarrow (\exists c : \alpha) c \in \text{claves}(d) \wedge_L \text{obtener}(d, c) = a$$

Ej. 2.

Un Videoclub es un negocio donde se tienen copias físicas de películas que pueden ser alquiladas. Los clientes alquilan las películas, se las llevan a su casa y luego las devuelven. Para cada película, existe una cantidad de días máxima en que esta puede ser tenida por una persona antes de devolverla. Si a un cliente se le vence el plazo para devolverla – se queda sin días para devolverla – automáticamente es desafiado del videoclub. Cuando esto sucede, el videoclub va a su casa para notificarlo y recuperar las películas que estaban alquiladas por el (ahora ex) cliente. Además, como el videoclub está recién empezando, solo posee una única copia de cada película en su catálogo.

Se pide completar el TAD presentado a continuación agregando los axiomas para los observadores **díasRestantesAlquiler** y **clientes**.

Se dispone de la función auxiliar **restarUno**: $\text{dicc}(\alpha, \text{nat}) \rightarrow \text{dicc}(\alpha, \text{nat})$ que resta uno a cada valor definido en el diccionario (y que se define si algún significado es igual a 0) y la función auxiliar **tieneValor**: $\text{dicc}(\alpha, \text{nat}) \times \text{nat} \rightarrow \text{bool}$ presentada en el ejercicio 1 (que pueden suponer definida aunque no lo hayan hecho).

TAD CLIENTE es string
TAD PELICULA es string

TAD VIDEOCLUB

igualdad observacional
(...)

géneros vc

observadores básicos

clientes	: vc	$\rightarrow \text{conj}(\text{cliente})$	
pelis	: vc	$\rightarrow \text{dicc}(\text{pelicula}, \text{nat})$	
díasRestantesAlquiler	: vc $\nu \times \text{cliente } c$	$\rightarrow \text{dicc}(\text{pelicula}, \text{nat})$	$\{c \in \text{clientes}(\nu)\}$
diaActual	: vc	$\rightarrow \text{nat}$	

generadores

inaugurarVC	: conj(cliente) $c \times \text{dicc}(\text{pelicula}, \text{nat})$ ps	$\rightarrow \text{vc}$	$\{\neg(\exists p \in \text{claves}(\text{ps}) \wedge_L \text{obtener}(\text{ps}, p) = 0)\}$
alquilar	: vc $\nu \times \text{cliente } c \times \text{pelicula } p$	$\rightarrow \text{vc}$	$\{c \in \text{clientes}(\nu) \wedge p \in \text{claves}(\text{peliculas}(\nu)) \wedge \neg(\exists c' \in \text{clientes}(\nu) \wedge_L \text{def?}(p, \text{díasRestantesAlquiler}(\nu, \{c'\}))\}$

devolver	: $vc \times cliente \times película \times p$	$\rightarrow vc$
		$\{c \in clientes(v) \wedge def?(p, diasRestantesAlquiler(v, c))\}$
pasarDia	: vc	$\rightarrow vc$
axiomas (extracto)		
$\forall v: videoclub, \forall c, c2: cliente, \forall p: película, \forall ps: conj(película), \forall cs: conj(cliente)$		
diaActual(inaugurarVC(cs, ps))		$\equiv 0$
diaActual(alquilar(v, c, p))		$\equiv diaActual(v)$
diaActual(devolver(v, c, p))		$\equiv diaActual(v)$
diaActual(pasarDia(v))		$\equiv diaActual(v) + 1$
diasRestantesAlquiler(inaugurarVC(cs, ps), c2)		$\equiv \dots$
diasRestantesAlquiler(alquilar(v, c, p), c2)		$\equiv \dots$
diasRestantesAlquiler(devolver(v, c, p), c2)		$\equiv \dots$
diasRestantesAlquiler(pasarDia(v), c2)		$\equiv \dots$
clientes(inaugurarVC(cs, ps))		$\equiv \dots$
clientes(alquilar(v, c, p))		$\equiv \dots$
clientes(devolver(v, c, p))		$\equiv \dots$
clientes(pasarDia(v))		$\equiv \dots$
Fin TAD		

Como se puede ver en el TAD, el Videoclub cuenta con las siguientes funciones:

- **clientes** devuelve los clientes actualmente afiliados al videoclub.
- **pelis** devuelve un diccionario con la cantidad de días máximos que una película puede ser alquilada.
- **diasRestantesAlquiler** devuelve la cantidad de días restantes para cada copia retirada por un cliente.
- **díaActual** devuelve cuantos días pasaron desde la apertura del videoclub.
- **inaugurarVC** crea un videoclub con un conjunto de clientes y un diccionario que indica la cantidad máxima de días de alquiler para cada película.
- **alquilar** permite a un afiliado alquilar una película
- **devolver** permite a un afiliado devolver una película

Ej. 3.

La famosa emisora televisiva HBoBo está lanzando su nuevo reality show, Juego de Calamares, que sigue en vivo y en directo la evolución de una colonia de calamares muy competitivos. El tanque será aislado al comenzar la serie, es decir, no ingresarán nuevos calamares durante el juego. Los calamares se organizan siguiendo a otros calamares que consideran sus líderes. Cada calamar puede tener (o no) varios seguidores, mientras que un calamar puede seguir a lo sumo a un único otro calamar. En cualquier momento, un calamar puede cambiar a qué calamar sigue. Durante las luchas de poder, es común que un calamar avergüenze a otro (le pinta la cara). En estos casos, todos los seguidores del calamar avergonzado pasan a seguir al avergonzador. El calamar avergonzado no puede con su orgullo y se retira del juego. HBoBo nos encarga especificar usando TADs el comportamiento de los clamares durante el Juego de Calamares. Se desea saber en todo momento quienes van ganando el Juego de Calamares; a tal efecto se considera que un calamar va ganando cuando sus seguidores (la cantidad total) son los más numerosos.

Dada la definición del problema presentado, definir un TAD JUEGODECALAMARES a partir de los siguientes pasos:

- (a) Listar en lenguaje natural qué aspectos son relevantes para diferenciar dos Juegos de Calamares.
- (b) Expresar esto en un conjunto de observadores y una definición de igualdad observacional. Comentar qué parte del enunciado expresa cada observador.
- (c) Definir un conjunto de generadores que permitan generar todos los valores relevantes del TAD. Comentar qué parte del enunciado expresa cada generador.
- (d) Axiomatizar el/los observadore/s que indican las relaciones de seguimiento entre los calamares.

$\text{tieneValor} : \text{dicc}(\alpha, \text{nat}) \times \text{nat} \rightarrow \text{bool}$

$\text{tieneValorAux} : \text{dicc}(\alpha, \text{nat}) \times \text{conj}(\alpha) \text{ as } \times \text{nat} \rightarrow \text{bool}$
 $\{ \text{as} \subseteq \text{claves}(d) \}$

$\forall d : \text{dicc}(\alpha, \text{nat}), \forall \text{as} : \text{conj}(\alpha), \forall n : \text{nat}$

$\text{tieneValor}(d, n) \equiv \text{tieneValorAux}(d, \text{claves}(d), n) \checkmark$

$\text{tieneValorAux}(d, \text{as}, n) \equiv$

if vacío?(as) then.

false

else

obtener(dameUno(as), d) = obs n

$\vee \text{tieneValorAux}(d, \text{sinUno}(as), n) \checkmark$

Fi

diasRestantesAlquiler (inaugurarVC (cs, ps), cz) \equiv vacío ✓

diasRestantesAlquiler (alquilar (v, c, p), cz) \equiv

if c = obs cz then

definir (p, polis(v), diasRestantesAlquiler (v, c))

else "polis(v)" es un dict, debería haber sido "obtener(p, polis(v))"

diasRestantesAlquiler (v, cz)

fi

diasRestantesAlquiler (devolver (v, c, p), cz) \equiv

if c = obs cz then

borrar (p, diasRestantesAlquiler (v, c))

else

diasRestantesAlquiler (v, cz)

fi ✓

diasRestantesAlquiler (pasarDia (v), cz) \equiv

restarUno (diasRestantesAlquiler (v, cz)) ✓

clientes (inaugurarVC (cs, ps)) \equiv cs ✓

clientes (alquilar (v, c, p)) \equiv clientes (v) ✓

clientes (devolver (v, c, p)) \equiv clientes (v) ✓

clientes (pasarDia (v)) \equiv desafiliarClientes (clientes (v), v) ✓

desafiliarClientes : conj (cliente) cs x VC v \rightarrow VC

$\{ cs \subseteq \text{clientes}(v) \}$

desafiliarClientes (cs, v) \equiv

if vacío? (cs) then

\emptyset

Debería ser 1, pues con 0

else

ya ~~la~~ hubiera sido sacada

if tieneValor (diasRestantesAlquiler(v, dameUno(cs)), 0) then

// se le venció el plazo de al menos una peli.

desafiliarClientes (sinUno(cs), v)

else

Aq (dameUno(cs), desafiliarClientes (sinUno(cs), v))

fi

✓

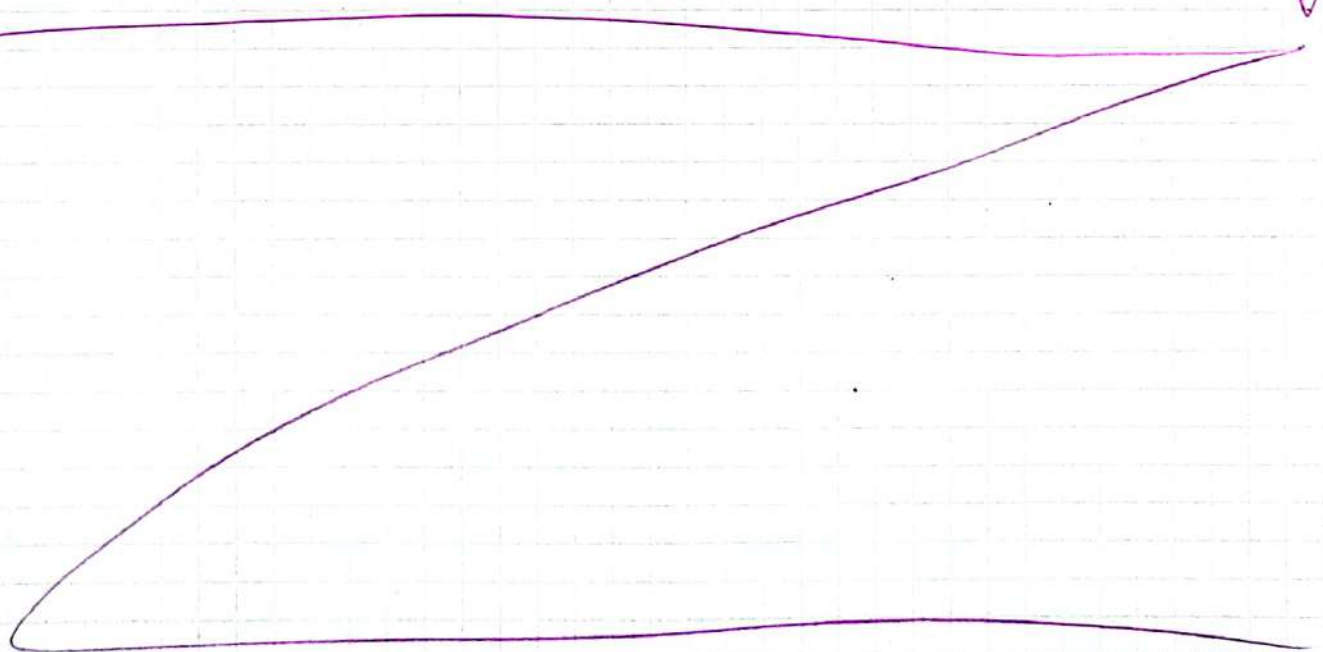
Fi



PARA DIFERENCIAR DOS JUEGOS SOLO NECESITAMOS SABER QUIÉNES ESTÁN AÚN EN EL JUEGO Y QUIÉN SIGUE A QUIÉN (LOS SEGUIDORES DE CADA CALAMAR). ✓

LA CANTIDAD DE SEGUIDORES DE CADA CALAMAR SE PUEDE CALCULAR A PARTIR DE SUS SEGUIDORES, POR LO TANTO NO NECESITAMOS ESTE DATO (COMO OBSERVADOR) PARA DIFERENCIAR DOS JUEGOS. ✓

A SU VEZ, COMO A HROBO NO LE INTERESA SABER EL HISTORIAL DE LUCHAS, NO NECESITAMOS LLEVAR REGISTRO DE QUIÉN LUCHÓ CON QUIÉN. ESTO SIGNIFICA QUE EN NUESTRO MODELO, DOS JUEGOS PUEDEN SER (OBSERVACIONALMENTE) IGUALES A PESAR DE HABERSE GENERADO DE FORMAS DISTINTAS. ✓ ES DECIR, CON DISTINTAS LUCHAS, DISTINTA SECUENCIA DE QUIÉN FUE SIGUIENDO A QUIÉN, E INCLUSO HABERSE INICIADO EL JUEGO CON GRUPOS DISTINTOS DE CALAMARES (AUNQUE EN ESTE CASO NECESITAMOS QUE HAYA AL MENOS UN CALAMAR EN COMÚN ENTRE LOS GRUPOS). ✓



OBSERVADORES BÁSICOS

calamares : juego \rightarrow conj(calamar) ✓

seguidores : calamar c x juego $j \rightarrow$ conj(calamar) ✓
 $\{ c \in \text{calamares}(j) \}$

IGUALDAD OBSERVACIONAL

$(\forall j_1, j_2 : \text{juego}) (j_1 =_{\text{obs}} j_2 \Leftrightarrow ($
 $\text{calamares}(j_1) =_{\text{obs}} \text{calamares}(j_2)$

$\wedge (\forall c : \text{calamar}) (c \in \text{calamares}(j_1) \Rightarrow_L ($
 $\text{seguidores}(c, j_1) =_{\text{obs}} \text{seguidores}(c, j_2)$

)

))

GENERADORES

comenzar: $\text{conj}(\text{calamar}) \text{ cs} \rightarrow \text{juego} \quad \{ \#(\text{cs}) > 0 \}$ ✓

Seguir: $\text{calamar } c \times \text{calamar } p \times \text{juego } j \rightarrow \text{juego}$

$\{ c \neq \text{obs } p \wedge \{c, p\} \subseteq \text{calamares}(j) \}$ ✓ $\wedge c \notin \text{seguidores}(p, j)$ ✓

avergonzar: $\text{calamar } c \times \text{calamar } p \times \text{juego } j \rightarrow \text{juego}$

$\{ c \neq \text{obs } p \wedge \{c, p\} \subseteq \text{calamares}(j) \}$ ✓ ✓

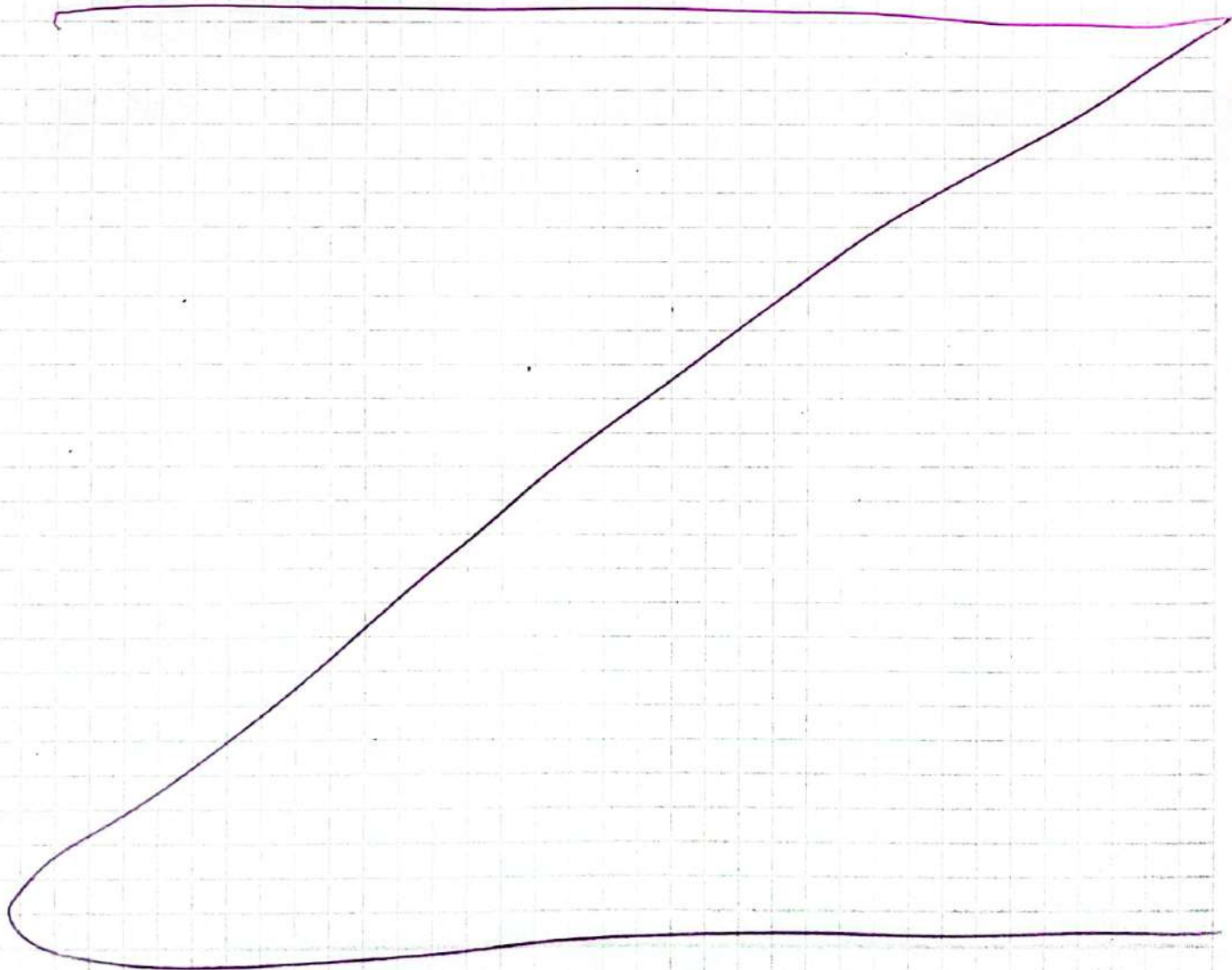
OTRAS OPERACIONES

ganadores: $\text{juego} \rightarrow \text{conj}(\text{calamar})$

// No consideramos a ganadores como un observador básico

// porque se puede axiomatizar a partir de los seguidores de

// cada calamar. ✓



AXIOMAS

$\forall j$: juego, $\forall c, c_2, p$: calamar, $\forall cs$: conj(calamar)

$\text{seguidores}(c, \text{comenzar}(cs)) \equiv \emptyset \checkmark$

$\text{seguidores}(c, \text{seguir}(c_2, p, j)) \equiv$

if $c = \text{obs } p$ then

$\text{Ag}(c_2, \text{seguidores}(c, j)) \checkmark$

else

// No sabemos si c_2 es seguidor de c , pero sí sabemos que

// a partir de ahora sigue a p , así que no puede ser

// seguidor de ningún otro calamar. *Está bien. xq el operador*

$\text{seguidores}(c, j) - \{c_2\} \checkmark$

fi

$\text{seguidores}(c, \text{avergonzar}(c_2, p, j)) \equiv$

if $c = \text{obs } c_2$ then

// como c_2 avergonzó a p , ahora todos los seguidores de p

// pasan a ser seguidores de c_2 (además de los que ya tenía)

$\text{seguidores}(c, j) \cup \text{seguidores}(p, j)$

// si bien p ya no está más en juego, en la instancia j

// previo al avergonzamiento sí lo está. Por eso podemos

// preguntar por los seguidores de p en j . *Exacto, había que hacerlo.*

else

Si $c_2 \in \text{segs}(p, j) \Rightarrow$ hay que sacar a c_2 de la
 $\text{seguidores}(c, j)$ *unión, xq c_2 no puede seguirse a sí mismo.*

fi

Además si $p \in \text{segs}(c_2, j) \Rightarrow$ hay que sacar a p de la unión,
xq si no va a haber un seguidor que quedo fuera del
juego; lo cual está malo

$\text{calamares}(\text{comenzar}(cs)) \equiv cs$ ✓

$\text{calamares}(\text{seguir}(c, p, j)) \equiv \text{calamares}(j)$ ✓

$\text{calamares}(\text{avergonzar}(c, p, j)) \equiv \text{calamares}(j) - \{p\}$ ✓

$\text{ganadores}(j) \equiv \text{ganadoresAux}(\text{calamares}(j), j)$ ✓

$\text{ganadoresAux} : \text{conj}(\text{calamar}) \text{ } cs \times \text{juego } j \rightarrow \text{conj}(\text{calamar})$
 $\{cs \subseteq \text{calamares}(j)\}$

$\text{ganadoresAux}(cs, j) \equiv$ No era necesario hacerlo, pero se valora positivamente haberlo hecho.

if vacío?(cs) then

∅

else

if #(seguidores(dameUno(cs), j)) = obs maxSeguidores(calamares(j), j) then

Ag(dameUno(cs), ganadoresAux(sinUno(cs), j))

else

ganadoresAux(sinUno(cs), j)

fi

fi

$\text{maxSeguidores} : \text{conj}(\text{calamar}) \text{ } cs \times \text{juego } j \rightarrow \text{nat} \{cs \subseteq \text{calamares}(j)\}$

$\text{maxSeguidores}(cs, j) \equiv$ if vacío?(cs) then 0 else

if #(seguidores(dameUno(cs), j)) > maxSeguidores(sinUno(cs), j) then

#(seguidores(dameUno(cs), j))

else

maxSeguidores(sinUno(cs), j)

fi

fi